UNIVERSITY OF CALIFORNIA
Santa Barbara

# Porting SPICE to iOS and Loading Netlists via QR Code

A Thesis submitted in partial satisfaction of the
requirements for the degree

Master of Science

in

Electrical and Computer Engineering

by

David Scott Gross

Committee in charge:
Professor K.T. Tim Cheng, Chair
Professor Bob York
Professor Umesh Mishra

June 2014

The thesis of David Scott Gross is approved.

_____
Bob York

_____
Umesh Mishra

_____
K.T. Tim Cheng, Committee Chair

June 2014

**Porting SPICE to iOS and Loading Netlists via QR Code**



by

**David Scott Gross**

## Curriculum Vitae

David Scott Gross

### EDUCATION

Bachelor of Science in Engineering Physics, University of California, San Diego, June 2012

Master of Science in Electrical and Computer Engineering, University of California, Santa Barbara, June 2014 (expected).

### PROFESSIONAL EMPLOYMENT

June - July 2013, Engineering Intern, Finisar Corporation, Fremont, CA

September 2012 – March 2014, Teaching Assistant, University of California, Santa Barbara

June 2009 – October 2011, Research Assistant, Center for Astrophysics & Space Sciences, University of California, San Diego

July – August 2011, COSMOS at UCSD, Cluster Assistant, University of California, San Diego

April – June 2010, San Diego Supercomputer Center, Assistant System Administrator

## ACKNOWLEDGEMENTS

Studying and working as a teaching assistant at UC Santa Barbara has been an exceptionally fulfilling experience. I had the opportunity to assist in teaching quantum mechanics, semiconductor physics, digital design, electromagnetic theory, and computer interfacing. The experiences and problem solving methods I have learned here will impact me in my engineering for the rest of my life.

I would like to acknowledge the DEF CON (https://www.defcon.org) worldwide hacker community and specifically the San Diego chapter of 2600 (http://2600.com). These groups have heavily influenced me to tinker and think creatively. DEF CON 19 and 20 were instrumental in laying the foundation for me to innovate with QR codes and within the iOS platform.

I would like to thank my close friend Robert Turner and SourceForge user Dietmar Warning for help cross-compiling SPICE to the ARM architecture. I would like to thank Val de Veyra for her help guiding me through this Master's program. Finally, I would like to thank my girlfriend Sireesha and my family for all of their support and understanding.

## ABSTRACT

A new iOS circuit simulation app has been developed, called “QRCircuit.” The app is designed for simulating undergraduate or graduate level amplifier configurations, filters, rectifiers, etc. It skips the schematic-based design approach to creating SPICE netlists typically seen in desktop applications and instead uses native Cocoa iOS GUI elements to enable rapid circuit prototyping and simulation. This allows the app to function more as a "calculator" and makes good use of the limited screen size of the iPhone™ and iPad™. While you can create a custom circuit netlist from scratch within the app, it has been designed from the start to provide a novel way to load netlists into it. On qrcircuit.com a repository of common netlists is provided for teaching purposes. Each circuit netlist has an accompanying QR code which goes along with it. Simply scanning the QR code in the app will load the netlist in. You can then edit, add, or remove any circuit elements or element models to your liking. It is anticipated that these QR codes will eventually be used in textbooks, PDFs, and websites to make these mediums more interactive. It is the goal of the author to eventually host every major type of circuit taught in an undergraduate or graduate curriculum this way.

## INTRODUCTION & MOTIVATION

The era of mobile computing has ushered in new tools and platforms that provide for incredible computational power at your finger tips. The iOS platform in particular, being the most senior and stable in this field, has seen a huge influx of developers. Berkeley SPICE [1] has a large following in the desktop world, and only a few developers have taken a shot at making this relatively complex software package available on a mobile device. While it is obvious that designing very complex integrated circuits on the small screen of an iPhone or iPad is practically impossible, there is a niche for simulation of most every circuit archetype seen in an undergraduate or graduate curriculum. The few existing iOS and Android apps for circuit simulation (EveryCircuit [2], Spicy Schematics [3], and iCircuit [4]) have their problems for serious electrical engineering students:

1. Some are not native and require the "cloud" or an Internet connection to run SPICE remotely and then download the results to the app. This solution does not scale, and simulation speed is limited by network speed. In addition, if the remote server goes down, the app becomes unusable.
2. None of the existing apps focus on physical understanding, despite being weakly marketed as such. The electrical engineering design process of physical understanding → experimental measurement → model derivation → circuit implementation → results and analysis of derived quantities is not evident in any of these apps. No node table print outs are accessible. It's not possible to take the derivative of an output vector or otherwise investigate the results. Component model properties such as oxide thickness, sheet resistance, etc. are inaccessible,

and only the most trivial of models and netlists can be computed. The most complex SPICE app currently available, EveryCircuit, allows the user to change the width and length of the transistor channel, but ends there in terms of complexity.

3. All of the apps take the schematic capture based approach to constructing the circuit netlist. This is fine for simple RC circuits, for example, but becomes increasingly tedious after a few components are added. This is all due to the limited screen real estate afforded to a mobile device. The command line inspired interface of QRCircuit is much more rapid and liberating.

Indeed, even most desktop SPICE implementations are not designed with the student in mind, are overly complex for achieving basic circuit insight, and require licensing and remote servers.

## APP DESCRIPTION

The new app developed, QRCircuit, attempts to solve these problems by taking on a radically different user interface based on native Cocoa Touch iOS elements. Ngspice [5], the open source successor of Spice3f5, has been cross-compiled to the latest generation of armv6, armv7, armv7s, and arm64-based mobile devices. Since SPICE is written entirely in C, it was straight forward to combine the SPICE data models with Objective C GUI code. The modern iPhone and iPad provides for computational power similar to desktop computers of only a few years ago, making it perfect for SPICE simulations.

The app provides full support for saving and recalling netlists, adding / removing / editing components and their respective models, investigating all node voltage and

currents, and plotting the simulation results and derived quantities. Screenshots of the app can be found in the Appendix. The app concentrates on the simulation of analog devices as the equations governing them are typically more complex than digital circuits at the undergraduate or graduate level. The typical analog circuit seen in the targeted learning environment does not contain many components (<20) and this means the mobile device has no disadvantages when compared to the classic desktop SPICE counterpart. In short, there exists a need for students to have a good "sanity-check" app to supplement their coursework.

QRCircuit is written entirely in Objective C and most of the GUI elements come from the standard Cocoa Touch library. The Node Table makes use of the open source Mutual Mobile SpreadsheetView [6] software package. SPICE itself is included in the app as a static library and is imported via a C header file, "sharedspice.h". The ngspice shared library API is used to access the native ngspice data structures via Objective C code references. "Help" functionality is provided at various points in the app to aid the user with data input. All component model parameters are specifically enumerated and defined (both their description and units) from within the app.

The app is called "QRCircuit" due to the development of a QR code-based standard for quick-loading SPICE netlists into the app for simulation (Figure 1). If this standard catches on, it could be the important link for hard-copy or PDF circuit textbooks that wish to be interactive. The circuit content could be closely controlled by the instructor. With a QR code representation of the circuit netlist, there is no time spent drawing the netlist on the mobile platform. The student can dive straight into the actual analysis.

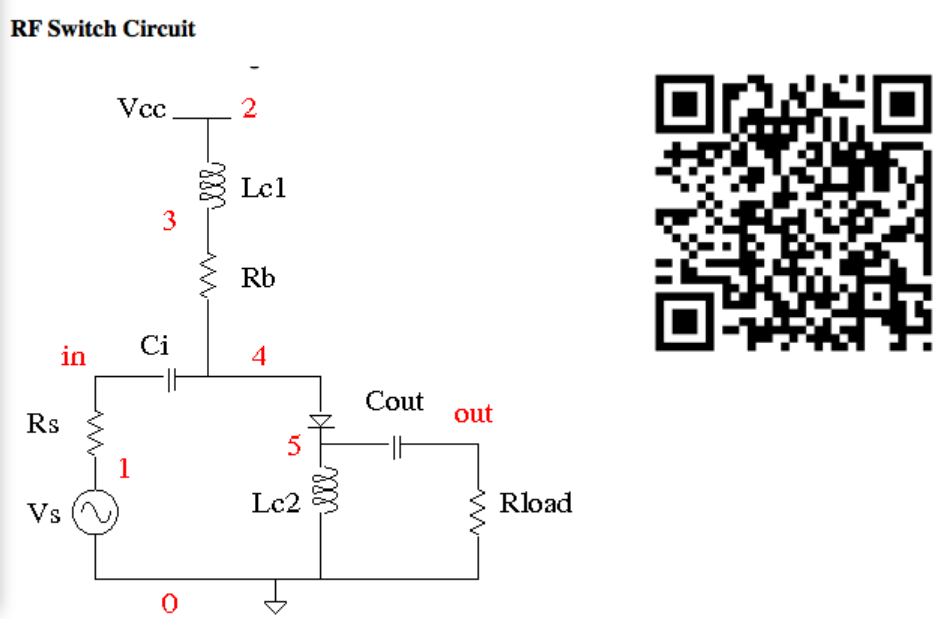



*Figure 1: Example schematic netlist / QR code scheme.*

The open-source zbar [7] barcode reader software package was cross-compiled to the various ARM architectures and used to scan and decode the QR code. A Perl script making use of the open-source Imager::QRCode module [8] was developed for custom QR code generation. Two different QR code standards were experimented with:

1. *A standard that requires a network connection.* This is the standard that was eventually settled on. A QR code is generated that has a qrcircuit.com hyperlink embedded in it. This hyperlink points to a text file containing the netlist. The user scans the QR code, the app makes a call to qrcircuit.com and retrieves the small text file netlist. This netlist is then parsed and validated by the app. This allows very complex netlists to be embedded in a small QR code. With the advent of Internet connectivity virtually everywhere, this approach has more pros than cons.

2. *A standard that does not require a network connection.* This was prototyped but ultimately deemed impractical. Only so many characters can be embedded in a QR code before it becomes too large, and therefore unreadable by the iPhone or iPad's camera. A couple approaches were investigated to compress the netlist so that it could fit into a QR code reasonably. Some Objective C code was even written to handle scanning of multiple QR codes in a serial fashion. While this could work well for small netlists (<10 components with limited model parameters), the results were not that impressive and this standard was tossed out.

QRCircuit handles simulation of the following analog devices currently: resistors, semiconductor resistors, capacitors, semiconductor capacitors, inductors, inductor models, coupled inductors, independent voltage / current sources (pulse, sinusoidal, exponential), linear-dependent sources (VCCS, VCVS, CCCS, CCVS), diodes, BJTs, JFETs, MESFETs, and MOSFETs. It currently supports small-signal AC analysis, DC transfer analysis, operating point analysis, and transient analysis at temperatures of the user's choosing.

One or two output vectors can be selected to analyze after the simulation results are obtained. This allows the user to investigate relative voltages between any pair of circuit nodes, rather than just between a circuit node and ground. The app currently supports converting to dB, absolute value, making negative, and derivative operations for any vector.

## FUTURE WORK

There is still much work to be done before the general release of the app on the iOS App Store. The app presented in this thesis has been a sole effort of the author over the course of the past eight months.

While some time was spent developing a recursive algorithm for reliably turning a SPICE netlist into a visually-appealing schematic, this problem is complex and could not be developed in time for this thesis. With a dedicated effort this should be possible. This alone could be a big development. There is not any quality free software that does this in the desktop or mobile world. Such software would be very useful in debugging netlists and ensuring correct input. The closest open-source project, Netlist Viewer [9], that attempts to do this is very basic, incomplete, and uses the antiquated Qt framework.

There is no native plotting library on iOS and the open source project imported for this purpose, CorePlot [10], leaves a lot to be desired. The axis scaling and labeling in particular on CorePlot is difficult to use. Some code redesign needs to be done to properly handle dB plots / Bode plots. In addition, a “hold on” feature needs to be implemented so that multiple data vectors can be shown on the same plot. This “hold on” feature was prototyped, but due to scaling issues, it became difficult to achieve a generalized solution. With more time, this should be possible. Many circuits have already been simulated via the app thanks to example netlists available online [11] [12] [13]. The results from these simulations were successfully compared against known results. The current plotting routines used in the app are stable, but there is always a need for more testing. It's also possible that a dedicated effort of cross-compiling GNUPLOT [14] for the various ARM architectures might be more fruitful long-term. GNUPLOT is the most stable and

well-supported plotting library for scientific applications and is also written in C.

Expanding to more supported circuit components will eventually be necessary. Eventually QRCircuit will support sources of the following types: piece-wise linear, single frequency FM, amplitude modulated (AM), transient noise, random voltage, external voltage or current input, and arbitrary phase. Transmission line and true analog model support (gain, summer, multiplier, divider, etc.) are essential and will also be added. In later releases, support for pole-zero analysis, small-signal distortion analysis, sensitivity analysis, noise analysis, periodic steady state analysis, and Fourier analysis will also be incorporated.

In a similar fashion to the "Circuit Library," a "Model Library" would also be a useful feature to add to allow the same component models to be used across different netlists. This can be accomplished via copy + paste in the app already, but a dedicated a Model Library would be sleeker.

Beta-testers will be needed throughout the summer to work out all the bugs and support new features. Please contact davidgross18@gmail.com if you are interested in helping out. The latest project developments can be found at http://qrcircuit.com.

## REFERENCES


1. *The Spice Page.* http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/

2. *EveryCircuit – Learn, Make, and Teach Electronics.* 2014. http://www.everycircuit.com

3. *Spicy Schematics – Modern Circuit Design for mobile and web.* 2011-2013. http://www.ischematics.com

4. Krueger, Frank A. *iCircuit.* 2010-2012. http://icircuitapp.com

5. *ngspice – Mixed Mode / Mixed Level Circuit Simulator*. 2011-2014. http://ngspice.sourceforge.net

6. Lacey, Jeff. Harwood, Kevin. *Mutual Mobile Spreadsheet View*. 2013. https://github.com/mutualmobile/MMSpreadsheetView

7. Brown, Jeff. *Zbar bar code reader.* 2007-2010. http://zbar.sourceforge.net/iphone/

8. Kurihara, Yoshiki. *Imager::QRCode.* 2011. http://search.cpan.org/~kurihara/Imager-QRCode/lib/Imager/QRCode.pm

9. Montorsi, Francesco. *Netlist Viewer.* https://sourceforge.net/projects/netlistviewer/

10. *core-plot*. https://code.google.com/p/core-plot/

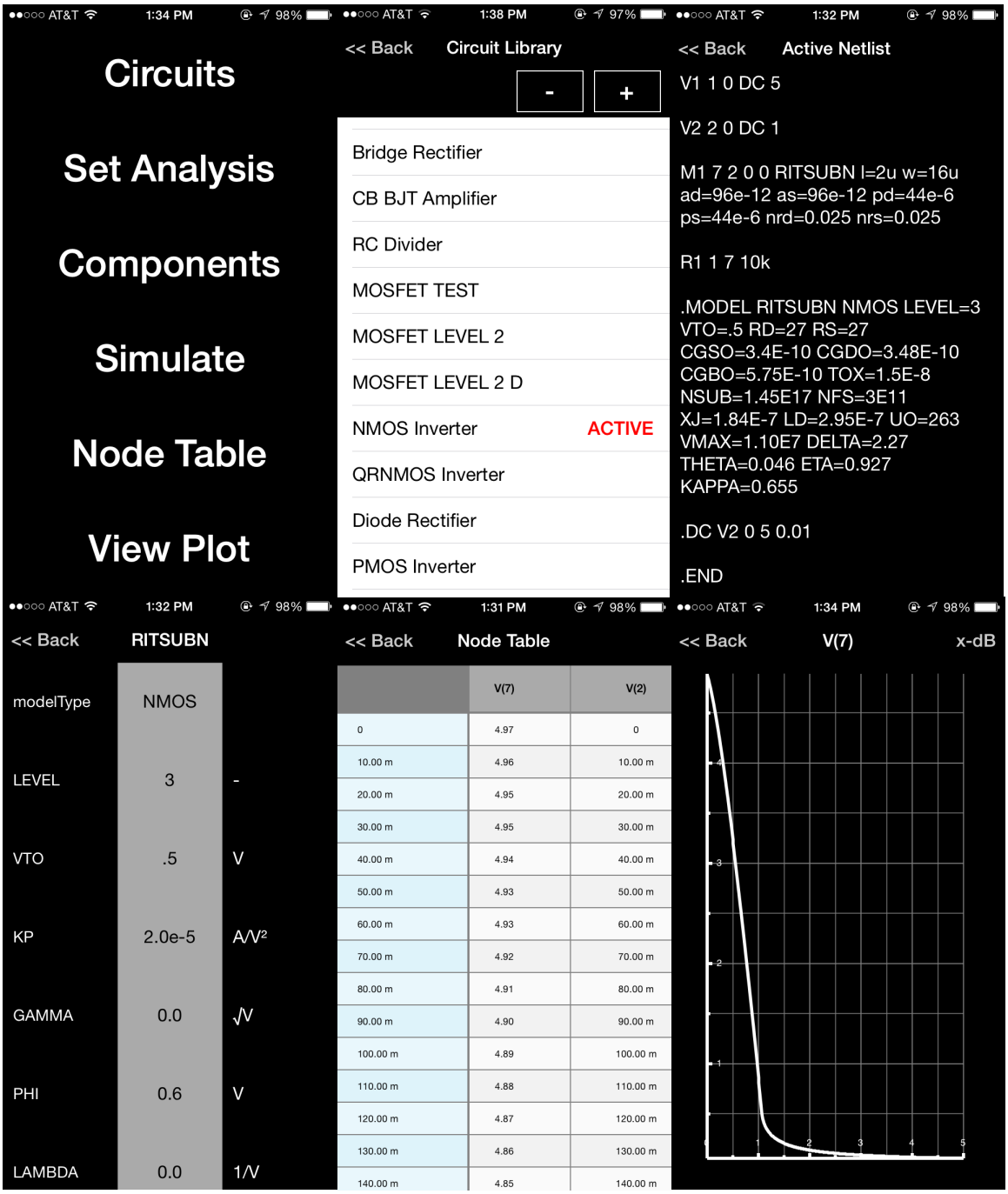
11. Christoffersen, Carlos. *Spice Tutorial 1.3.0 documentation*. 2012-2013. http://vision.lakeheadu.ca/eng4136/spice/intro.html

12. *All About Circuits – Example Circuits and Netlists.* 2003-2012. http://www.allaboutcircuits.com/vol_5/chpt_7/8.html

13. Fuller, Lynn. *SPICE Examples Using OrCAD PSPICE and WINSPICE.* 2011. http://people.rit.edu/lffeee/SPICE_Examples_OrCAD_WinSPICE_Fuller.pdf

14. *gnuplot homepage.* 2014. http://www.gnuplot.info

## APPENDIX (APP SCREENSHOTS)




*Assorted QRCircuit Screenshots (from left to right, top to bottom): 1. Main Menu, 2. Circuit Library, 3. Active Netlist View, 4. NMOS Model Parameter Details Viewer / Editor, 4. Node Table, 5. Output plot of a NMOS Inverter w/ Resistor load after a DC sweep from 0 to 5 V.*